Lab 8 Documentation

Elizabeth Huang
EE 104
Dr. Christopher Pham

Abstract

This lab uses Python to build a CNN in order to classify different animals and objects. To practice GUI programming another game was made.

Objective

The objective of this lab is to apply neural network modeling concepts and Python methods learned in lecture to increase the accuracy of a CNN classifying animals and objects. Another fun objective of the lab is to continue practicing GUI programming with the creation of another game, Balloon Flight!

References

Dr. Christopher Pham's Module 8 Lectures & Code

Coding Games In Python

& its Python Games Resource Pack: Chapter 8 Balloon Flight

Instructions/Documentation

Do ensure that the following python modules are installed and imported:

For CNN

(Was ran on Colab, not needed to install anything, but in case you need to install:)

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
import numpy as np
```

For the game, Balloon Flight!

pip install pgzero on command line

```
import pgzrun
from pgzero.builtins import Actor
from random import randint
```

CNN Classifier

Added data augmentation layers, multitude of other layers and dropout layer of 30%

```
[4] # augmentation
data_augmentation = keras.Sequential(
  [
    layers.RandomFlip("horizontal",
                      input_shape=(32,
                                   32,
                                   3)),
    layers.RandomRotation(0.1),
  ]
)

model = models.Sequential()
model.add(data_augmentation)
model.add(layers.Conv2D(64, (3, 3), input_shape=(32, 32, 3)))
model.add(layers.Conv2D(64, (3, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(256, (3, 3), activation='relu'))
model.add(layers.Conv2D(256, (3, 3), activation='relu'))
```

```
model.add(layers.Dropout(0.3))
model.add(layers.Flatten())
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dense(10))
```

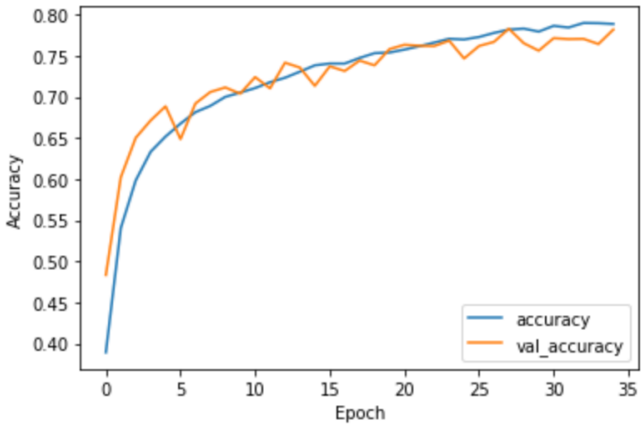
Evaluate the model

```
[13] plt.plot(history.history['accuracy'], label='accuracy')
     plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
     plt.xlabel('Epoch')
     plt.ylabel('Accuracy')
     plt.legend(loc='lower right')

     test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)
```

2s

```
313/313 - 1s - loss: 1.3372 - accuracy: 0.6215 - 1s/epoch - 3ms/step
```



val_accuracy: 0.7903

```
airplane_url = "https://upload.wikimedia.org/wikipedia/commons/thumb/f/f0/Another_Airplane%21_%28467
airplane_path = tf.keras.utils.get_file('Blue_airplane', origin=airplane_url)

img = tf.keras.utils.load_img(
    airplane_path, target_size=(32, 32)
)
img_array = tf.keras.utils.img_to_array(img)
img_array = tf.expand_dims(img_array, 0) # Create a batch

predictions = model.predict(img_array)
score = tf.nn.softmax(predictions[0])

print(
    "This image most likely belongs to {} with a {:.2f} percent confidence."
    .format(class_names[np.argmax(score)], 100 * np.max(score))
)
```

```
This image most likely belongs to airplane with a 100.00 percent confidence.
```

```
automobile_url = "https://sniteartmuseum.nd.edu/assets/166204/original/ferrari.jpg"
automobile_path = tf.keras.utils.get_file('Red_automobile', origin=automobile_url)

img = tf.keras.utils.load_img(
    automobile_path, target_size=(32, 32)
)
img_array = tf.keras.utils.img_to_array(img)
img_array = tf.expand_dims(img_array, 0) # Create a batch

predictions = model.predict(img_array)
score = tf.nn.softmax(predictions[0])

print(
    "This image most likely belongs to {} with a {:.2f} percent confidence."
    .format(class_names[np.argmax(score)], 100 * np.max(score))
)
```

```
Downloading data from https://sniteartmuseum.nd.edu/assets/166204/original/ferrari.jpg
98304/96948 [==============================] - 0s 2us/step
106496/96948 [==============================] - 0s 2us/step
This image most likely belongs to automobile with a 100.00 percent confidence.
```

```
[22] horse_url = "https://upload.wikimedia.org/wikipedia/commons/0/03/American_quarter_horse.jpg"
horse_path = tf.keras.utils.get_file('Brown_horse', origin=horse_url)

img = tf.keras.utils.load_img(
    horse_path, target_size=(32, 32)
)
img_array = tf.keras.utils.img_to_array(img)
img_array = tf.expand_dims(img_array, 0) # Create a batch

predictions = model.predict(img_array)
score = tf.nn.softmax(predictions[0])

print(
    "This image most likely belongs to {} with a {:.2f} percent confidence."
    .format(class_names[np.argmax(score)], 100 * np.max(score))
)
```

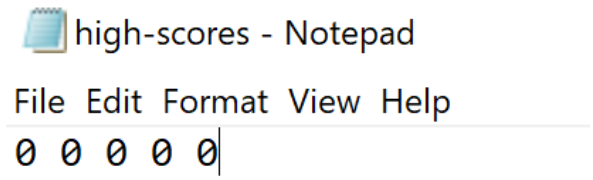
```
Downloading data from https://upload.wikimedia.org/wikipedia/commons/0/03/American_quarter_horse.jpg
2547712/2543247 [==============================] - 0s 0us/step
2555904/2543247 [==============================] - 0s 0us/step
This image most likely belongs to horse with a 100.00 percent confidence.
```

Game Development – Balloon Flight!

It's a beautiful day for a balloon flight! Steer your hot air balloon to keep you and your passengers safe in this game. 5 new features were added to the initial sample code provided by Chapter 8 Balloon Flight tutorial from *Coding Games in Python*:

- More High Scores: Stats for high scores have been adjusted to allow top 5 high scores.
- Speed Up: The bird obstacle has been sped up by 2.
- Different Way to Score: Initial scoring process was inefficient. Once the user's balloon passes an obstacle, it immediately increments the score instead of waiting for the obstacle to leave the screen.
- Space Out: The tree obstacle and house obstacles are spaced out. The bird is already spaced out due to its much faster speed.
- A Little More Room: For more mobility, the balloon's top is now allowed to go off screen a little (but not too much).

To implement More High Scores, adjust the high-scores text file to 5 numbers instead of 3.



Don't forget to make sure the filename's pathway is correctly directed to the high-scores file in the folder.

```
39  def update_high_scores():
40      global score, scores
41      filename = (r'C:\Users\lingt\OneDrive\Documents\EE104\Lab 8\Chapter 8 Balloon Flight\high-scores.txt')
```

Additional flags to handle identifying when balloon passes the three different obstacles were added in line 34 through 37 of the code.

```
11  # define actors
12  balloon = Actor('balloon')
13  balloon.pos = 400, 300
14
15  bird = Actor('bird-up')
16  bird.pos = randint(800, 1600), randint(10, 200)
17
18  house = Actor('house')
19  house.pos = randint(800, 1600), 460
20
21  tree = Actor('tree')
22  tree.pos = randint(800, 1600), 450
23
24  # define flags
25  bird_up = True
26  up = False
27  game_over = False
28  score = 0
29  number_of_updates = 0
30
31  # for score keeping
32  scores = []
33
34  # to flag midpoints for incrementing score
35  birdPassedMidpoint = False
36  housePassedMidpoint = False
37  treePassedMidpoint = False
```

All other changes to sample code is in update(). The global variables are updated to include the added flags. The if statement starting in line 110 shows how an obstacle passing the balloon's flag is handled. If an obstacle passes the balloon without being hit (is less than 400), then it is marked True and the score is incremented. Line 113 shows that the bird is sped up by 2 (initially the variable was 4, now it is 6). When an obstacle gets off the screen, the flag is reset to False (example in line 120). The tree and house are spaced out with the if statement starting in line 145. The if statement checks if the tree has been assigned a value within += 200 of the house. It spaces out the tree by 500 from the house if it is too close to the house. Lastly, line 149 was adjusted to allow some mobility room for the top of the balloon to go off the screen a little without making the player lose.

```python
def update():
    global game_over, score, number_of_updates
    # flags for objects passing the balloon/midpoint
    global birdPassedMidpoint, housePassedMidpoint, treePassedMidpoint
    if not game_over:
        if not up:
            balloon.y += GRAVITY_STRENGTH  # gravity
        if bird.x > 0:
            # case statement to incrment score if balloon passes bird
            if bird.x < 400 and birdPassedMidpoint == False:
                score += 1
                birdPassedMidpoint = True #flags midpoint
            bird.x -= 6 # + 2 speed
            if number_of_updates == 9:
                flap()
                number_of_updates = 0
            else:
                number_of_updates += 1
        else:
            birdPassedMidpoint = False #resets midpoint flag
            bird.x = randint(800, 1600)
            bird.y = randint(10, 200)
            number_of_updates = 0

        if house.right > 0:
            # case statement to incrment score if balloon passes house
            if house.x < 400 and housePassedMidpoint == False:
                score += 1
                housePassedMidpoint = True #flags midpoint
            house.x -= 2
        else:
            housePassedMidpoint = False #resets midpoint flag
            house.x = randint(800, 1600)

        if tree.right > 0:
            # case statement to incrment score if balloon passes tree
            if tree.x < 400 and treePassedMidpoint == False:
                score += 1
                treePassedMidpoint = True #flags midpoint
            tree.x -= 2
        else:
            treePassedMidpoint = False #resets midpoint flag
            tree.x = randint(800, 1600)
            # if tree is within 200 from house
            if tree.x < house.x + 200 and tree.x > house.x - 200:
                tree.x = house.x + 500 # space out tree from house by 500

        #adjusted so top of balloon can go off screen a little for more mobility
        if balloon.top < -100 or balloon.bottom > 560:
            game_over = True
            update_high_scores()

        if (balloon.collidepoint(bird.x, bird.y) or
                balloon.collidepoint(house.x, house.y) or
                balloon.collidepoint(tree.x, tree.y)):
            game_over = True
            update_high_scores()

```